Package 'frscore'

May 20, 2022

Title Functions for Calculating Fit-Robustness of CNA-Solutions

Version 0.2.0

Description Functions for automatically performing
a reanalysis series
on a data set using CNA, and for calculating the fit-robustness
of the resulting models, as described in
Parkkinen and Baumgartner (2021) <doi:10.1177/0049124120986200>.

License AGPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports dplyr, Rfast, magrittr, rlang

Depends R (>= 3.5), cna (>= 3.2.0)

Suggests spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Veli-Pekka Parkkinen [aut, cre, cph],
Michael Baumgartner [aut, cph],
Mathias Ambuehl [ctb]

Maintainer Veli-Pekka Parkkinen <parkkinenv@gmail.com>

Repository CRAN

Date/Publication 2022-05-20 13:40:02 UTC

R topics documented:

d.error *Simulated data of sixteen cases with measurement error in one case*

Description

A simulated set of crisp-set configurational data that conforms to the structure A + B*C <-> E except for one case, 'c16', which simulates measurement error in that case, and including one irrelevant factor "D".

Usage

```
d.error
```

Format

An object of class `data.frame` with 16 rows and 5 columns.

frscore *frscore*

Description

Calculate fit-robustness scores for a set of `cna` solutions/models

Usage

```
frscore(
  sols,
  scoretype = c("full", "supermodel", "submodel"),
  normalize = c("truemax", "idealmax", "none"),
  maxsols = 50,
  verbose = FALSE,
  print.all = FALSE
)
```

Arguments

`sols`	Character vector of class "stdAtomic" or "stdComplex" (as generated by `cna()`) that contains the solutions/models to be scored.
`scoretype`	Character vector specifying the scoring method: `"full"` (default; scoring is based on counting both sub- and supermodels), `"supermodel"` (count supermodels only), `"submodel"` (count submodels only).
`normalize`	Character vector that determines the method used in normalizing the scores. `"truemax"` (default) normalizes by the highest score among the elements of `sols`, such that the highest scoring solution types get score 1. `"idealmax"` normalizes by a theoretical maximum score (see Details).

maxsols	Integer determining the maximum number of unique solution types found in sols to be included in the scoring (see Details).
verbose	Logical; if TRUE, additional information about submodel relations among the unique solution types found in sols is printed. Defaults to FALSE.
print.all	Logical, controls the number of entries printed when printing the results. If TRUE, results are printed as when using the defaults of print.data.frame. If FALSE, 20 highest scoring solutions/models are printed.

Details

frscore() implements fit-robustness scoring as introduced in Parkkinen and Baumgartner (2021). The function calculates the fit-robustness scores of Boolean solutions/models output by the cna() function of the **cna** package. The solutions are given to frscore() as a character vector sols obtained by reanalyzing a data set repeatedly, e.g. with rean_cna(), using different consistency and coverage thresholds in each analysis.

When the argument scoretype is set to its default value "full", the score of each sols[i] is calculated by counting both the sub- and supermodel relations sols[i] has to the other elements of sols. Setting scoretype to "supermodel" or "submodel" forces the scoring to be based on, respectively, supermodel and submodel relations only. In the former setting, less complex models will tend to get higher fit-robustness scores because they tend to have more supermodels in sols, while the latter setting gives preference to more complex models, which have more submodels in sols on average.

The fit-robustness scores can be normalized in two ways. In the default setting normalize = "truemax", the score of each sols[i] is divided by the maximum score obtained by an element of sols. In case of normalize = "idealmax", the score is normalized not by an actually obtained maximum but by an idealized maximum, which is calculated by assuming that all solutions of equal complexity in sols are identical and that for every sols[i] of a given complexity, all less complex elements of sols are its submodels and all more complex elements of sols are its supermodels. When normalization is applied, the normalized score is shown in its own column 'norm.score' in the results. The raw scores are shown in the column 'score'.

If the size of the consistency and coverage interval scanned in the reanalysis series generating sols is large or there are many model ambiguities, sols may contain so many different types of solutions/models that robustness cannot be calculated for all of them in reasonable time. In that case, the argument maxsols allows for capping the number of solution types to be included in the scoring (defaults to 50). frscore() then selects the most frequent solutions/models in sols of each complexity level until maxsols is reached and only scores the thus selected elements of sols.

If the argument verbose is set to TRUE, frscore() also prints a list indicating for each sols[i] how many raw score points it receives from which elements of sols. The verbose list is ordered with decreasing fit robustness scores.

Value

A named list where the first element is a data frame containing the unique solution/model types and their scores. Rest of the elements contain additional information about the submodel relations among the unique solutions types and about how the function was called.

References

V.P. Parkkinen and M. Baumgartner (2021), "Robustness and Model Selection in Configurational Causal Modeling," *Sociological Methods and Research*, doi:10.1177/0049124120986200.

Basurto, Xavier. 2013. "Linking Multi-Level Governance to Local Common-Pool Resource Theory using Fuzzy-Set Qualitative Comparative Analysis: Insights from Twenty Years of Biodiversity Conservation in Costa Rica." *Global Environmental Change* **23** (3):573-87.

See Also

rean_cna, cna

Examples

```
# Artificial data from Parkkinen and Baumgartner (2021)
sols1 <- rean_cna(d.error, attempt = seq(1, 0.8, -0.1))
sols1 <- do.call(rbind, sols1)
frscore(sols1$condition)

# Real fuzzy-set data from Basurto (2013)

sols2 <- rean_cna(d.autonomy, type="fs", ordering = list("EM", "SP"),
         strict = TRUE, maxstep = c(3,3,9))
sols2 <- do.call(rbind, sols2)$condition  # there are 217 solutions
# At the default maxsols only 50 of those solutions are scored.
frscore(sols2)
# By increasing maxsols the number of solutions to be scored can be controlled.
frscore(sols2, maxsols = 100)

# Changing the normalization
frscore(sols2, normalize = "none")
frscore(sols2, normalize = "truemax")
frscore(sols2, normalize = "idealmax")

# Changing the scoring
frscore(sols2, scoretype = "supermodel")
frscore(sols2, scoretype = "submodel")

frscore(sols2, maxsols = 20, verbose = TRUE)
```

---

frscored_cna *frscored_cna*

Description

Perform a reanalysis series on a data set and calculate the fit-robustness scores of the resulting solutions/models

Usage

```
frscored_cna(
  x,
  fit.range = c(1, 0.7),
  granularity = 0.1,
  output = c("csf", "asf"),
  scoretype = c("full", "supermodel", "submodel"),
  normalize = c("truemax", "idealmax", "none"),
  verbose = FALSE,
  maxsols = 50,
  test.model = NULL,
  print.all = FALSE,
  ...
)
```

Arguments

x	A data.frame or configTable to be analyzed with cna(). In case of multi-value or fuzzy-set data, the data type must be indicated by type = "mv" and type = "fs", respectively.
fit.range	Numeric vector of length 2; determines the maximum and minimum values of the interval of consistency and coverage thresholds used in the reanalysis series. Defaults to c(1, 0.7).
granularity	Numeric scalar; consistency and coverage are varied by this value in the reanalysis series. Defaults to 0.1.
output	Character vector that determines whether csfs or asfs are returned; "csf" (default) returns csfs, "asf" asfs.
scoretype	Character vector specifying the scoring method: "full" (default; scoring is based on counting both sub- and supermodels), "supermodel" (count supermodels only), "submodel" (count submodels only).
normalize	Character vector that determines the method used in normalizing the scores. "truemax" (default) normalizes by the highest score among the elements of sols, such that the highest scoring solution types get score 1. "idealmax" normalizes by a theoretical maximum score (see Details).
verbose	Logical; if TRUE, additional information about submodel relations among the unique solution types found in sols is printed. Defaults to FALSE.
maxsols	Integer determining the maximum number of unique solution types found in the reanalysis series to be included in the scoring (see Details).
test.model	Character vector that specifies a single candidate cna() solution/model whose fit-robustness score is calculated against the results of the reanalysis series.
print.all	Logical that controls the number of entries printed when printing the results. If TRUE, results are printed as when using the defaults of print.data.frame. If FALSE, 20 highest scoring solutions/models are printed.
...	Any arguments to be passed to cna() except con, cov or con.msc. The effect of argument what is overriden by output.

Details

`frscored_cna()` is a wrapper function that sequentially executes `rean_cna()` and `frscore()`, meaning it performs both computational phases of fit-robustness scoring as introduced in Parkkinen and Baumgartner (2021). In the first phase, the function conducts a reanalysis series on the input data `x` at all combinatorially possible combinations of fit thresholds that can be generated from the interval given by `fit.range` at increments given by `granularity` and collects all solutions/models in a set **M**. In the second phase, it calculates the fit-robustness scores of the atomic (asf) and/or complex (csf) solution formulas in **M**. The argument `output` allows for controlling whether csf or only asf are built, the latter normally being faster but less complete.

When the argument `scoretype` is set to its default value `"full"`, the score of each solution/model in **M** is calculated by counting both the sub- and supermodel relations it has to the other elements of **M**. Setting `scoretype` to `"supermodel"` or `"submodel"` forces the scoring to be based on, respectively, supermodel and submodel relations only. In the former setting, less complex models will tend to get higher fit-robustness scores because they tend to have more supermodels in **M**, while the latter setting gives preference to more complex models, which have more submodels in **M** on average.

The fit-robustness scores can be normalized in two ways. In the default setting `normalize = "truemax"`, the score of each solution/model is divided by the maximum score obtained by an element of **M**. In case of `normalize = "idealmax"`, the score is normalized not by an actually obtained maximum but by an idealized maximum, which is calculated by assuming that all solutions of equal complexity in **M** are identical and that for every model of a given complexity, all less complex elements of **M** are its submodels and all more complex elements of **M** are its supermodels.

If the argument `verbose` is set to `TRUE`, `frscored_cna()` also prints a list indicating for each solution/model how many raw score points it receives from which elements of **M**. The verbose list is ordered with decreasing fit robustness scores.

If the size of the consistency and coverage range scanned in the reanalysis series generating **M** is large or there are many model ambiguities, **M** may contain so many different types of solutions that robustness cannot be calculated for all of them in reasonable time. In that case, the argument `maxsols` allows for capping the number of solution types to be included in the scoring (defaults to 50). `frscored_cna()` then selects the most frequent solutions in **M** of each complexity level until `maxsols` is reached and only scores the thus selected elements of **M**.

If the user is interested in the robustness of one specific candidate model, that model can be given to `forscored_cna()` by the argument `test.model`. The result for that model will then be printed separately, provided the model is found in the reanalysis series, if not, the function stops.

Value

A list whose first element is a data frame that contains the model types returned from a reanalysis series of the input data, their details such as consistency and coverage, together with the unadjusted fit-robustness score of each model type shown in column 'score', and a normalized score in column 'norm.score' in case `normalize = "truemax"` or `normalize = "idealmax"`. The other elements contain additional information about the submodel relations among the unique solution types and about how the function was called.

References

P. Emmenegger (2011) "Job Security Regulations in Western Democracies: A Fuzzy Set Analysis." *European Journal of Political Research* 50(3):336-64.

C. Hartmann and J. Kemmerzell (2010) "Understanding Variations in Party Bans in Africa." *Democratization* 17(4):642-65. doi:10.1080/13510347.2010.491189.

V.P. Parkkinen and M. Baumgartner (2021), "Robustness and Model Selection in Configurational Causal Modeling," *Sociological Methods and Research*, doi:10.1177/0049124120986200.

See Also

frscore, rean_cna

Examples

```
# Robustness analysis from sect. 4 of Parkkinen and Baumgartner (2021)
frscored_cna(d.error, fit.range = c(1, 0.75), granularity = 0.05,
             ordering = list("E"), strict = TRUE)

# Multi-value data from Hartmann and Kemmerzell (2010)
frscored_cna(d.pban, type = "mv", fit.range = c(0.9, 0.7), granularity = 0.1,
                  normalize = "none", ordering = list("T", "PB"), strict = TRUE)

# Fuzzy-set data from Emmenegger (2011)
frscored_cna(d.jobsecurity, type = "fs", fit.range = c(0.9, 0.6), granularity = 0.05,
                  scoretype = "submodel", ordering = list("JSR"), strict = TRUE)

# Artificial data
dat <- data.frame(
  A = c(1,1,0,0,0,0,1,1),
  B = c(0,1,0,0,0,0,1,1),
  C = c(1,0,1,0,1,0,1,0),
  D = c(1,1,0,0,1,1,0,0),
  E = c(1,1,1,1,0,0,0,0))
frscored_cna(dat)
frscored_cna(dat, output = "asf")
frscored_cna(dat, maxsols = 10)
frscored_cna(dat, test.model = "(b*e+A*E<->D)*(B<->A)")
```

---

rean_cna *rean_cna*

Description

Perform a reanalysis series on a data set with cna() using all combinations of consistency and coverage threshold values in a given range of values

Usage

```
rean_cna(x, attempt = seq(1, 0.7, -0.1), ncsf = 20, output = c("csf", "asf"), ...)
```

Arguments

x	A data.frame or configTable to be analyzed with cna(). In case of multi-value or fuzzy-set data, the data type must be indicated by type = "mv" and type = "fs", respectively.
attempt	Numeric vector that contains the values from which combinations of consistency and coverage thresholds are formed, to be used in the analyses.
ncsf	Scalar integer that determines the maximum number of csfs (complex solutions formulas) calculated in a each cna() analysis, if output = "csf". Defaults to 20.
output	Character vector that determines whether csfs or asfs are returned; "csf" (default) returns csfs, "asf" asfs.
...	Any arguments to be passed to cna() except con, cov or con.msc. The effect of argument what is overriden by output.

Details

rean_cna() performs a reanalysis series of a data set x, which constitutes the first computational phase of fit-robustness scoring as introduced in Parkkinen and Baumgartner (2021). The series consists of cna() calls at all combinatorially possible consistency and coverage settings drawn from the vector attempt. If the output argument is set to its default value "csf", rean_cna() returns complex solutions formulas (csf), in case of "asf" only atomic solution formulas ("asf") are built, which is faster. The argument ncsf allows for controlling the number of csf to be built, if output = "csf".

Value

A list where each element is a data frame containing the results of a single analysis of the input data set with cna(), each using a different combination of consistency and coverage threshold values. These values are added to the output as extra columns 'cnacon' and 'cnacov'.

References

V.P. Parkkinen and M. Baumgartner (2021), "Robustness and Model Selection in Configurational Causal Modeling," *Sociological Methods and Research*, doi:10.1177/0049124120986200.

See Also

frscore, cna

Examples

```
# Crisp-set data
sols1 <- rean_cna(d.error, attempt = seq(1, 0.8, -0.1))
sols1 <- do.call(rbind, sols1)
sols1

# Multi-value data
sols2 <- rean_cna(d.pban, type = "mv", attempt = seq(0.9, 0.7, -0.1),
```

```
                  ordering = list("T", "PB"), strict = TRUE)
sols2 <- do.call(rbind, sols2)
sols2

# Fuzzy-set data
sols3 <- rean_cna(d.jobsecurity, type = "fs", attempt = seq(0.9, 0.6, -0.05),
              ordering = list("JSR"), strict = TRUE) # execution takes a couple of seconds
sols3 <- do.call(rbind, sols2)
sols3
```

Index